Anomaly Detection and Levels of Automation for AI-supported System Administration

Anton Gulenko, Odej Kao, Florian Schmidt

TU Berlin, 10587, Berlin, Germany
{firstname}.{lastname}@tu-berlin.de

Abstract. Artificial Intelligence for IT Operations (AIOps) describes the process of maintaining and operating large IT infrastructures using AI-supported methods and tools on different levels, e.g. automated anomaly detection and root cause analysis, remediation and optimization, as well as fully automated initiation of self-stabilizing activities. While the automation is mandatory due to the system complexity and the criticality of a QoS-bounded response, the measures compiled and deployed by the AI-controlled administration are not easily understandable or reproducible in all cases. Therefore, explainable actions taken by the automated systems are becoming a regulatory requirement for future IT infrastructures. In this paper we present a developed and deployed system named ZerOps as an example for the design of the corresponding architecture, tools, and methods. This system uses deep learning models and data analytics of monitoring data to detect and remediate anomalies.

Keywords: AIOps, Predictive fault tolerance, Anomaly detection

1 Introduction

The cloud, softwareization and virtualization trend, the increasing number of IoT applications with dynamically linked devices and the embedding in real-world environments drive the creation of large multilayered systems. Consequently, the system complexity is growing up to a level, where it is impossible for human operators to oversee and holistically manage the systems without additional support and automation. Uninterrupted services with guaranteed latencies and response times are however a mandatory prerequisite for many data-driven and autonomous applications. Therefore, downtimes and malfunctioning components may have a crucial impact. On the other hand, the software-defined technology on all layers of the infrastructure stack opens new possibilities to control not only the server landscape, but also the connected frontend devices and the communication paths. This optimization potential can be utilized to increase the dependability and the reliability of the overall system.

The large service providers are aware of the need for always-on, dependable services and already introduced additional intelligence to the IT-ecosystem, e.g. by employing network and site reliability engineers, by deploying automated tools for 24/7 monitoring, and AIOps platforms for load balancing, capacity planning, resource utilization,

storage management, and threat detection. This decision-making in combination with advanced virtualization techniques allows a significant progress regarding reliability, serviceability, and availability: checkpointing, migration, re-starts, scale-out/scale-down, re-routing, resource reservations, and other operations are much more flexible and easier to deploy, even independent from the underlying hardware infrastructure and not limited to a certain site.

The next piece of the puzzle aims at rapidly decreasing the reaction time, in case an urgent activity of a system administrator – e.g. due to performance problems (tuning), to component/system failures (outages, degraded performance), or due to security incidents – is necessary. All these examples describe situations where the system operates outside of the normal (expected or pre-defined) parameters. Thus, the system exposes an anomaly that must be registered, recognized, and remediated before it leads to a component or a system failure. Many anomalies are handled straightforwardly, for example component outages and interrupted services have a massive impact and trigger a lot of indicators to be noticed by the administrators. More sophisticated malfunctions, such as stressed CPUs, leaks of main memory or storage space, increased network latency are far more complex to be detected, taking the number of involved cores, networks, and software systems into account. Usually, these effects propagate horizontally and/or vertically leading at some point of time – if not detected and remediated – to an interrupted service noticed by an administrator. Finding the cause for the observed symptoms and then repairing the cause requires a valuable time in a worst case visible to the customer due to a service downtime or due to a significantly reduced QoS. Therefore, much effort was spent in recent years to speed-up this process by developing and deploying advanced mechanisms for monitoring the systems, aggregating and analyzing the data, detecting the root cause, and finally remediating it.

The remaining of the paper is organized as follows. We will review the existing taxonomies on levels of automation in section 2 and present a set of rules for AIOps. Section 3 illustrates the feasibility of the proposed rules in typical application scenarios. Section 4 describes a sample AIOps system using stream processing for anomaly detection and remediation.

2 Background

The rules of automation for an AI-supported IT infrastructure management should help human IT operators to find a common ground for discussion of the AIOps features and tools and to clearly define the policies for the day-to-day operation using the benefits of the automated administration while simultaneously knowing and respecting the boundaries. Sheridan and Verplank [1] provided in 1978 one of the first listings of rules and levels of autonomous operation in the context of undersea teleoperators. The list contains ten rules for the computer and human:

Level 1: the operator does the task and turns it over to the computer to implement.
Level 2: the computer helps determining the options.
Level 3: the computer determines and suggests options. The operator can choose to

follow the recommendation.

Level 4: the computer selects the action and the operator decides, if it should or should not be done.

Level 5: the computer selects the action and implements it, if the operator approves the action.

Level 6: the computer selects the action and informs the operator in case the operator wants to cancel the action.

Level 7: the computer does the action and tells the human operator what it did.

Level 8: the computer does the action and reports only if the operator asks.

Level 9: the computer does the action when told and tells the human operator only if the computer decides the operator should be told.

Level 10: the computer does the action if it decides it should be done. The computer tells the human operator, only if it decides the operator should be told.

These rules describe a wide spectrum of human-controlled, computer-assisted, computer-controlled, up to an autonomous mode of operation. Endsley developed in [2,3] a similar hierarchy for expert systems to supplement human decision-making. Another combination of rules for automation and architecture was provided by IBM in 2002 coining the term of autonomic computing [4,5]. The five levels range from a basic, fully manually level up to an autonomic level with an interplay between autonomic components and business rules/policies without a human intervention. The levels named managed, predictive, and adaptive do not specify the applied technology, so they are transferable into the current world of AIOps. A valuable extension is given on the alignment with the business goals, as the authors described the impact of autonomic computing on the processes, tools, skills, and benchmarks. One of the core contributions in the autonomic computing initiative is the introduction of the self-X paradigm. Originally, four properties – self-configuring, self-healing, self-optimizing, and self-protecting – were presented, each of them addressing a capability of an autonomic system to deal with an induced modification. The need for change can be caused by manually initiated demands or by monitoring the environment and responding to degraded performance, security attacks, or system malfunctions.

We use in following the levels of automation proposed by Sheridan and Verplank for describing automation in IT infrastructures.

3 Application Scenarios

The AIOps include a large set of typical activities such as runtime optimization, remediation/recovery after malfunctioning, updates, security upgrades, and sizing (scale-up/-down) tasks. Applying a fully automated operation is possible in all these cases, however the cost of false decisions and consequences varies significantly.

Runtime optimization

The runtime optimization and sizing aim at keeping the QoS parameters within predefined bounds, e.g. the response time is less than x seconds, the CPU load is lower

than y%, or the disk free capacity does not fall below z%. The typical activities such as adding/removing nodes, replicating the database, re-sizing storage volumes may have impact on the overall system performance, but usually not on the functionality reducing the risk significantly. Therefore, the level 5 automation is the standard in the majority of the productive infrastructures. The level 6 automation may be useful for optimization cases triggered by unusual circumstances, e.g. by DDoS attacks. The optimization can switch the strategy from up-scaling to reconfiguring the balancing frontends to block suspected queries / IP ranges.

Updates

System updates are frequent operations on all layers of the IT infrastructure. Mobile devices are typically updated automatically without user interaction (level 5). A step towards level 6 is implemented by initiating an automatic rollback to the last known working setup in case of update failure. Server and other core components are challenging to update due to the individual configuration and the necessary interaction. Level 4 is therefore the standard approach. Level 5 is reached in combination with a testing and deployment pipeline, where the updates are verified prior migration to the production lines. Deployment frameworks (e.g. Kubernetes) allow a gradual update of multiple components with fast rollback.

Reliability, availability, security

A guaranteed continuous service within pre-defined QoS parameters is the main prerequisite for IoT applications. Thus, reliability and availability need as much real-time automation as possible. Automating the anomaly detection in IT infrastructures using AI-methods (deep learning, time series analysis) is currently a hot topic in the academia and industry. These approaches correspond to at least level 5 automation or– depending on the deployed policy – to level 6. Automatic remediation is examined as well, where the system evaluates existing recovery and remediation options and applies the selected operation until the anomaly disappears or the set of feasible options is empty. Even higher levels are targeted by the idea of self-healing and self-stabilizing systems, where the system is able to generate a feasible remediation workflow autonomously, but there are no implemented solutions yet.

4 ZerOps system for anomaly detection and remediation

The AIOps demonstrator ZerOps provides a self-stabilization pipeline consisting of components for monitoring, streaming data analytics, and of an AI-controlled remediation / recovery engine [6]. The implementation is based on OpenStack and is experimentally evaluated using two real-world domains: a core network NFV scenario based on the virtual IP multimedia subsystem Clearwater and a streaming edge cloud service with low latency demands following a real-time messaging protocol. The detection of the anomalies before causing a failure is a key requirement for AIOps systems. Inspired by Kephart and Chess [10] work about autonomic computing, Figure 1 shows the self-sustainable AIOps loop. Monitoring data are continuously collected from the monitored

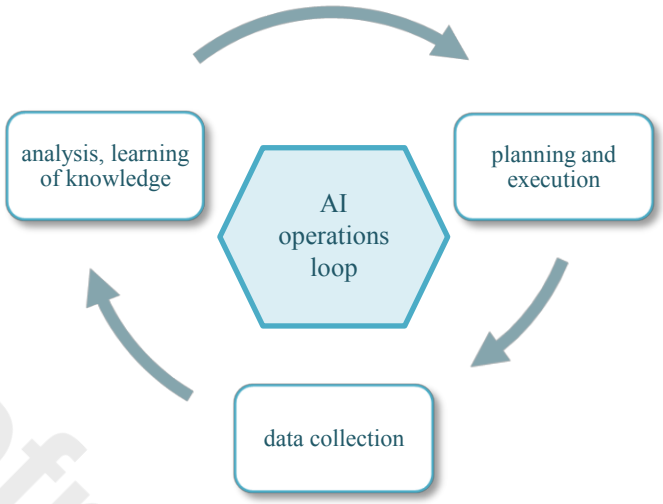

Fig. 1. Steps of a self-sustainable AIOps loop [10].

system, then analyzed in order to extract and maintain a knowledge base, and finally used to select, plan and execute remediation workflows in anomaly situations.

Ghosh et al. [7] confine anomalies to a fuzzy deterioration zone between the normal and failed state of individual system components. Thus, anomalies can manifest themselves as changes of individual or combined system metrics like increased latency, decreased throughput, unusual CPU or disk I/O utilization patterns and many others. The ZerOps system continuously monitors a large number of indicators to determine whether the monitored infrastructure components are exposing abnormal system metric levels or patterns. The collected data include time series metrics from multiple sources across the technological stack. Operating systems deliver real-time resource utilization data for the entire host, or for individual processes. Both physical and virtual hosts deliver these generic data, which are completely independent of the actual workload of the system. Other sources of data are more application and network protocol specific, like the HTTP response latency or number of incoming requests per second. In order to reduce network bandwidth overhead and the data analysis latency, the data are analyzed and pre-aggregated directly at the source, while keeping the computational overhead within pre-defined bounds.

The data are analyzed using machine learning methods. A fully supervised approach relies on injection and learning of possible anomalies and thus can be implemented straightforwardly. However, anomaly injections within a provider system might harm the system and the results in real-world settings are not satisfactory due to the presence of noise and the necessity for operators to introduce all known anomalies [8]. Thus, collecting training data within a relevant production environment remains as challenge. Besides supervised machine learning approaches, which require data of all anomalies, the applied unsupervised approach requires only data of the running system without artificially injected anomalies in order to detect the “normal” state and identify all data

points that do not fit into the learned model as abnormal. Thus, the requirement for labeled training is abandoned and we solely assume that anomalies occur far less frequently than normal operation corresponding to the level 5 expectations.

Level 6 automation requires in addition a fully unsupervised machine learning model adaptation of thresholds or cluster borders to the current system load situation [11], [12]. This can be achieved by online adaption to concept drifts of the machine learning models.

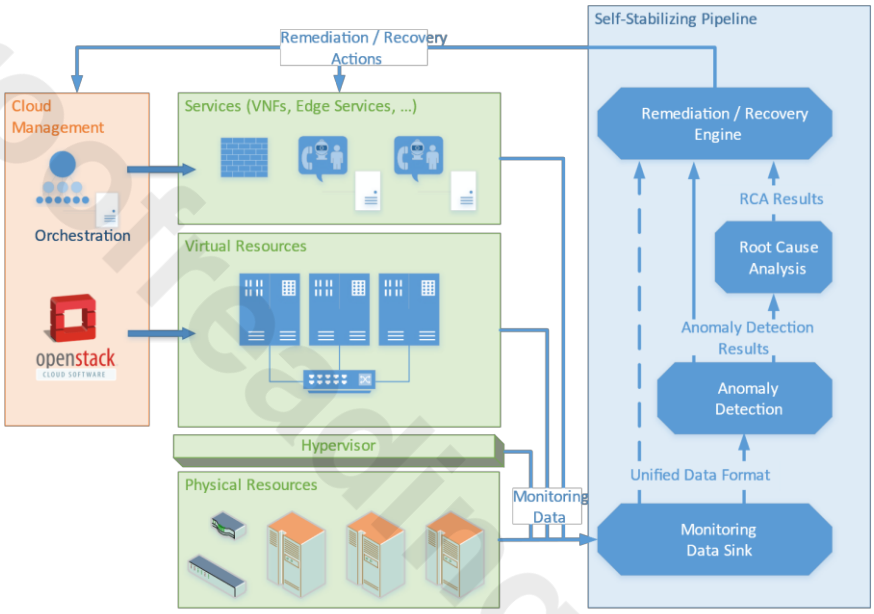

Fig. 2. Architecture of an OpenStack cloud infrastructure with a self-stabilizing pipeline.

Detected anomalies can be either the source or the symptom of horizontally or vertically propagating failures. Therefore, in a largely distributed infrastructure with high dependency between system parts, the identification of the root cause (root cause analysis – RCA) during an anomaly scenario is essential [9]. We propose to utilize results from the anomaly detection models to either directly determine root causes or to obtain valuable indications to reduce the set of possible root cause nodes to enable reduction of complexity and time efficient analysis for the next analysis steps.

The proposed rules of automation above level 2 require the system to either suggest or autonomously execute countermeasures in case of detected anomalies. Thus, the remediation engine requests monitoring data to apply time series pattern matching methods (see Figure 2). Combining the results of anomaly detection, RCA, and pattern matching enables the system to determine whether a similar situation was previously encountered. In this case, the system can provide automatic selection of options to an expert user based on applied solutions in the past. This includes also the decision whether any historic counteraction provided positive value to solve the underlying problem.

5 Summary

The AIOps platforms aim at supporting system operators to holistically control large, complex IT infrastructures with attached QoS requirements found in many current and future domains. The monitoring components, methods for data analytics and machine learning as well as the knowledge repositories with patterns for detecting and remediating anomalies allow the development and deployment of tools and frameworks for largely autonomous control of infrastructures and implementation of the self-X vision established in the last decade. We presented a developed demonstrator utilizing unsupervised machine learning to detect anomalies, identify the root cause, and finally deploy remediation workflows to stabilize the overall system and prevent the anomaly turning into failure.

The future work will be focused on verifying the scope of the defined rules in different IT domains. Thus, aiming to introduce additional precision, if necessary. Moreover, we are experimentally evaluating the demonstrators in real-world environments in order to adapt the models to the characteristics of productive infrastructures.

References

1. Sheridan, T. B., & Verplank, W. L. “Human and computer control of undersea teleoperators”. Cambridge, Mass: Massachusetts Institute of Technology, Man-Machine Systems Laboratory (1978). www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA057655 PDF,
2. Endsley, M. “The application of human factors to the development of expert systems for advanced cockpits”, Proceedings of the Human Factors Society, 1388 ± 1392. M 1987
3. Endsley, M. "Level of automation effects on performance, situation awareness and workload in a dy-namic control task." Ergonomics 42.3 (1999): 462-492.
4. Ganek, Alan G., and Thomas A. Corbi. "The dawning of the autonomic computing era." IBM systems Journal 42.1 (2003): 5-18.
5. Computing, Autonomic. "An architectural blueprint for autonomic computing." IBM White Paper 31 (2006): 1-6.
6. Gulenko, Anton, et al. "A system architecture for real-time anomaly detection in large-scale NFV systems." Procedia Computer Science 94 (2016): 491-496.
7. Ghosh, Debanjan, et al. "Self-healing systems—survey and synthesis." Decision support systems 42.4 (2007): 2164-2185.
8. Gulenko, A., Schmidt, F., Kao, O. “Evaluating machine learning algorithms for anomaly detection in clouds”. In IEEE International Conference on Big Data (2016) pp. 2716-2721.
9. Solé, Marc et al. “Survey on Models and Techniques for Root-Cause Analysis.” CoRRabs/1701.08546 (2017)
10. Kephart, Jeffrey O., and David M. Chess. “The vision of autonomic computing.” Computer 1(2003): 41-50.
11. Hundman, K., et al “Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding”. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (2018), pp. 387-395.
12. Schmidt, F., Gulenko, A., Kao, O. “IFTM-Unsupervised Anomaly Detection for Virtualized Network Function Services”. In 2018 IEEE International Conference on Web Services (ICWS) (2018) pp. 187-194.